

# INTEGRATING SOFTWARE STACKS

With **BuildStream 2.0** and the **Remote Execution API**

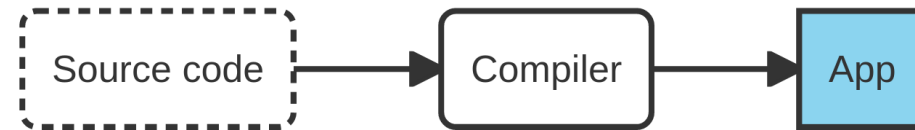
Sam Thursfield

OSSEU 2022



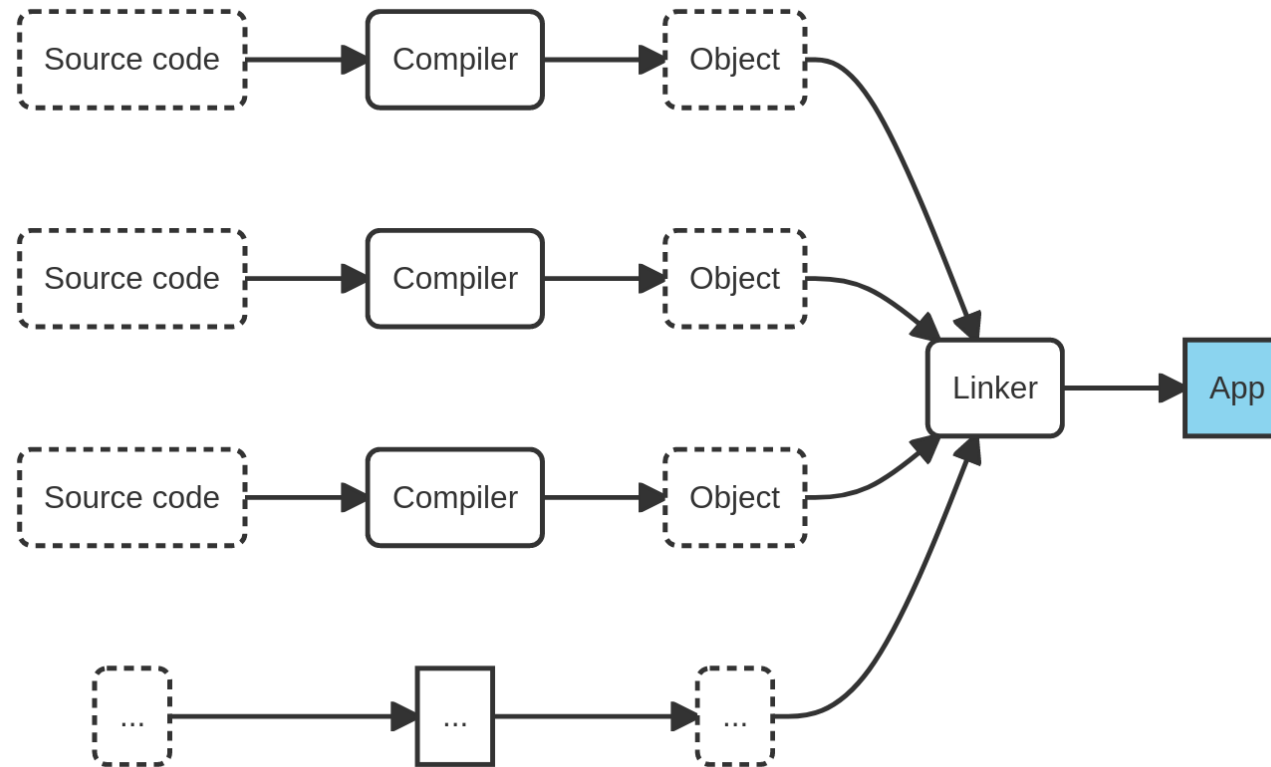
# THE BASIC PROBLEM

Compiling source code to binaries



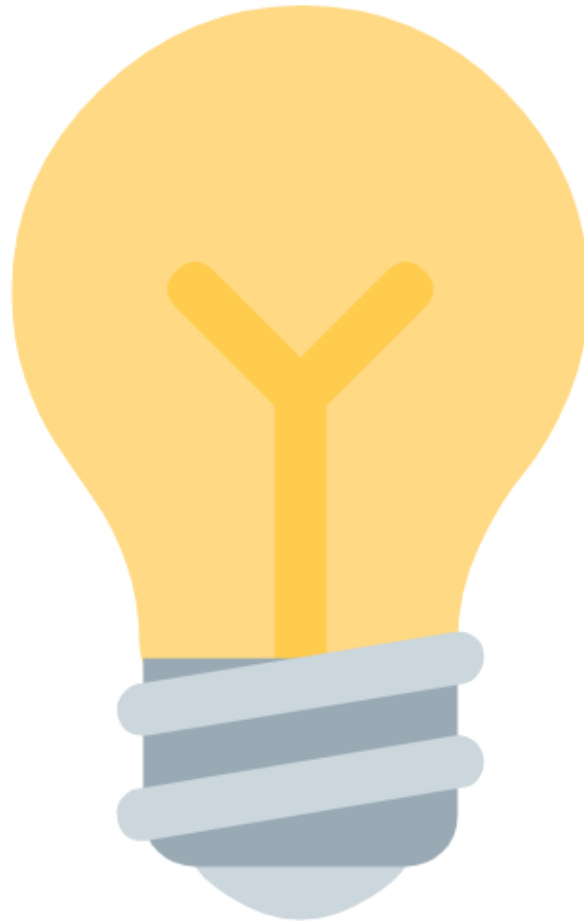
# THE BASIC PROBLEM

Compiling source code to binaries



# A BRIEF HISTORY OF INTEGRATION TOOLS

In the beginning was: **Make**



# A BRIEF HISTORY OF INTEGRATION TOOLS

In the beginning was: **Make**

	Compiler	Make
Dependency tracking	None	File level

# A BRIEF HISTORY OF INTEGRATION TOOLS

In the beginning was: **Make**

	Compiler	Make
Dependency tracking	None	File level
Multiple repos	No	No
Repeatable builds	No	No

# A BRIEF HISTORY OF INTEGRATION TOOLS

Then came the **packaging tools**...

	<b>Compiler</b>	<b>Make</b>	<b>Packaging tools</b>
Dependency tracking	None	File-level	Package-level
Multiple repos	No	No	Yes
Repeatable builds	No	No	Mostly

# A BRIEF HISTORY OF INTEGRATION TOOLS

- **Meta-build tools** (Buildroot, BuildStream, Yocto)
- **Containers** (Docker, Podman, Flatpak)
- **Continuous integration**
- and more...

	Compiler	Build tools	Integration tools
Dependency tracking	None	File-level	Package-level
Multiple repos	No	No	Yes
Repeatable builds	No	No	Mostly

# INTEGRATING AT SCALE IN 2022

One tool to "rule them all" ...?



# One tool to "rule them all" ...?

Image by [Selrond](#), used under Creative Commons BY-NC-SA license

# INTEGRATING AT SCALE IN 2022

*Make each program do one  
thing well.*

— Doug McIlroy, The Bell System Technical Journal, July-August 1978

# INTEGRATING AT SCALE IN 2022

*Make each program do one  
thing well.*

— Doug McIlroy, The Bell System Technical Journal, July-August 1978

**How can our integration tools work together?**

# THE REMOTE EXECUTION API

Introduced in 2017, by team at Google working on Bazel.

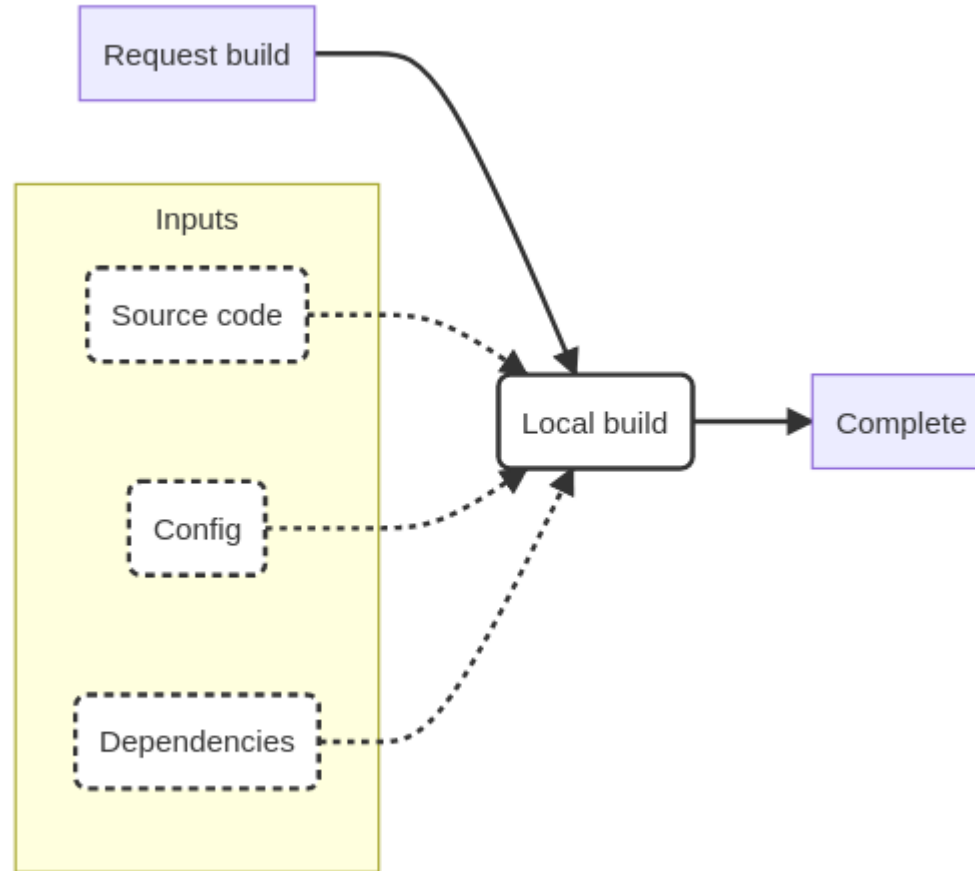
Designed around:

- Content addressable storage for directories and files
- An execution request interface
- A **result cache**

Building blocks for any integration pipeline you can think of.

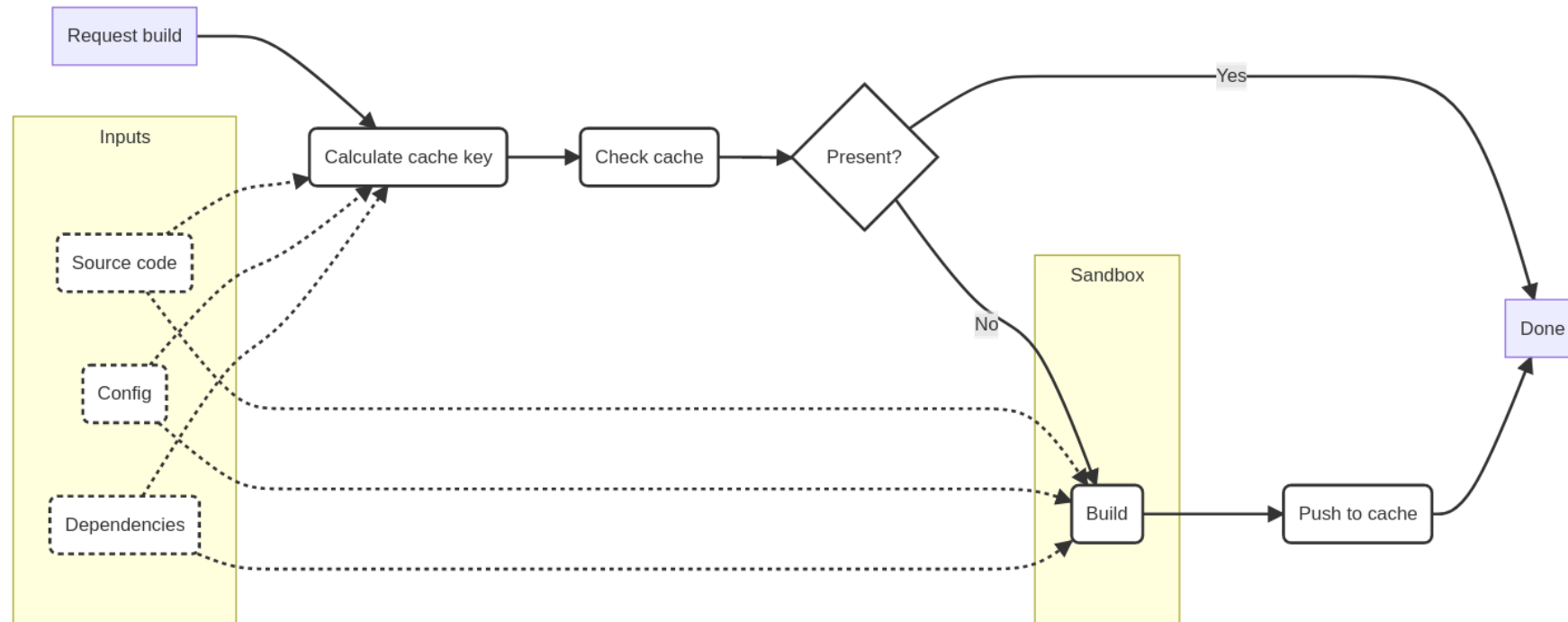
# THE REMOTE EXECUTION API

Traditional build tools



# THE REMOTE EXECUTION API

## Content-addressed caching



For cache to be trusted and shared, cache key must capture all factors that can affect build output.

The cache and the builder can be local, or remote.

# THE REMOTE EXECUTION API

## New Remote Execution API Draft

Subscribe ☐ 

146 views



**Alexis Hunt**

to bazel-...@googlegroups.com

18 Apr 2017, 18:20:06   

Hello Bazel community!

A few months ago, we circulated a draft API for remote execution & caching in Bazel. This was implemented as a prototype, and based on our work with this, we've prepared a revised draft of the API.

The basic functionality is still the same:


- A content-addressable storage for storing directory trees and files.
- An execution request interface.
- A result cache.

Many of the details have been changed based on feedback and where we thought shortcomings in the original design were. We've also endeavoured to bring the API more in line with Google API standards.

The design document (including the API spec) is available at <https://goo.gl/TB49ED>. Please offer any feedback or suggestions you have in the document itself. Once there is consensus on the API, we will work to update the prototype in Bazel to match.

Thanks


# THE REMOTE EXECUTION API

 **bazelbuild** / **remote-apis** Public

👁 Watch 65 🍴 Fork 84 ☆ Star 187

<> Code 🔍 Issues 37 🔗 Pull requests 7 🎬 Actions 📁 Projects 🛡 Security ...

🔗 main Go to file Add file Code

 **Yannic** Make it explicit that there are no m... ... on 27 Jul 🕒 160

📁 .github/workflows	Fix go bindings generation (#182)	2 years ago
📁 build/bazel	Make it explicit that there are no ...	last month
📁 external	Migrating to Bazel 1.0. Fixes #101...	3 years ago
📁 hooks	Add LogStream (#148)	2 years ago
📁 internal	Move language-specific targets to...	2 months ago
📄 .bazelrc	Include comments in generated *...	2 years ago
📄 .gitignore	Add a quick and dirty build with b...	4 years ago
📄 AUTHORS	Initial commit: licensing and repos...	5 years ago
📄 BUILD.bazel	Rename top-level BUILD to BUIL...	3 years ago


## About

An API for caching and execution of actions on a remote system.

- 📖 Readme
- 📄 Apache-2.0 license
- ☆ 187 stars
- 👁 65 watching
- 🍴 84 forks

---

## Releases 2

 **v2.0.0** Latest  
on 4 Nov 2019

# THE REMOTE EXECUTION API

The screenshot shows the GitHub repository page for `bazelbuild/remote-apis`. The repository is public and has 65 watchers, 84 forks, and 187 stars. The navigation bar includes links for Code, Issues (37), Pull requests (7), Actions, Projects, and Security. The current branch is `main`, and the file path is `remote-apis/build/bazel/remote/execution/v2/`. The commit history table shows the following entries:

Commit	Message	Time
..		
cc	Move language-specific targets to subdirectories. (#2...	2 months ago
go	Move language-specific targets to subdirectories. (#2...	2 months ago
java	Move language-specific targets to subdirectories. (#2...	2 months ago
BUILD	Move language-specific targets to subdirectories. (#2...	2 months ago
platform.md	platform.md: Add RISC-V ISA names <code>rv32g</code> and <code>rv...</code>	2 months ago
remote_execution.pb.go	Regenerate .pb.go (#224)	4 months ago
remote_execution.proto	Make it explicit that there are no method-specific error...	last month

# REAPI CLIENTS

≡ README.md

There are a number of clients and services using these APIs, they are listed below.





## Clients

These tools use the Remote Execution API to distribute builds to workers.

- [Bazel](#)
- [BuildStream](#)
- [Goma Server](#)
- [Pants](#)
- [Please](#)
- [Recc](#)





# REAPI CLIENTS

All these tools support remote caching and distributed builds via the Remote Execution API.

	<b>Name</b>	<b>Dependency tracking</b>	<b>Repeatable builds</b>	<b>Multiple repos</b>
Compiler tools	Goma Server	-	-	-
	Recc	-	-	-
Build tools	 Bazel	File-level	Partly <sup>1</sup>	Partly
	 Pants	File-level	Partly <sup>1</sup>	No
	 Please	File-level	Yes	No
Integration tools	 BuildStream	Element-level	Yes	Yes

# REAPI CLIENTS





All these tools support remote caching and distributed builds via the Remote Execution API.

	Name	Dependency tracking	Repeatable builds	Multiple repos
Compiler tools	Goma Server	-	-	-
	Recc	-	-	-
Build tools	 Bazel	File-level	Partly <sup>1</sup>	Partly
	 Pants	File-level	Partly <sup>1</sup>	No
	 Please	File-level	Yes	No
Integration tools	 BuildStream	Element-level	Yes	Yes

Compiler tools: replacements for ccache + distcc.

# REAPI CLIENTS

All these tools support remote caching and distributed builds via the Remote Execution API.

	Name	Dependency tracking	Repeatable builds	Multiple repos
Compiler tools	Goma Server	-	-	-
	Recc	-	-	-
Build tools	 Bazel	File-level	Partly <sup>1</sup>	Partly
	 Pants	File-level	Partly <sup>1</sup>	No
	 Please	File-level	Yes	No
Integration tools	 BuildStream	Element-level	Yes	Yes





Compiler tools: replacements for ccache + distcc.

Build tools: replacements for Make

1. Bazel and Pants sandboxing doesn't hide host environment in any way.

# REAPI CLIENTS

All these tools support remote caching and distributed builds via the Remote Execution API.

	Name	Dependency tracking	Repeatable builds	Multiple repos
Compiler tools	Goma Server	-	-	-
	Recc	-	-	-
Build tools	 Bazel	File-level	Partly <sup>1</sup>	Partly
	 Pants	File-level	Partly <sup>1</sup>	No
	 Please	File-level	Yes	No
Integration tools	 BuildStream	Element-level	Yes	Yes

Compiler tools: replacements for ccache + distcc.

Build tools: replacements for Make

1. Bazel and Pants sandboxing doesn't hide host environment in any way.

Integration tools: designed for integrating multiple components, such as packages.

# THINGS TO BE AWARE OF

- Bazel isn't magic
  - Adapting an existing project to Bazel means rewriting its entire build system
  - Requires a BUILD file for each 3rd party dependency
  - Sandbox can be leaky.
  - Hard to extend beyond built-in languages, no plugin mechanism
  - Hard to land changes upstream
  - CLI is not pleasant to use
- BuildStream isn't magic
  - Designed more for integrators and less for developers
  - If making frequent changes to slow-to-build elements, be prepared to wait
  - Sandboxing helps ensure repeatable builds, but nothing is foolproof

# REAPI INFRASTRUCTURE

Name	Language	Storage	Execution
<a href="#">Bazel Buildfarm</a>	Java	disk, S3, shard, ...	local
<a href="#">BuildBarn</a>	Go	disk, S3, shard	local
<a href="#">BuildGrid</a>	Python/C++	disk, Redis, S3, shard, ...	local, bwrap, OCI, ...
<a href="#">Please-servers</a>	Go	Google Cloud Services	local
<a href="#">Scoot</a>	Go	-	local

I recommend BuildBarn or BuildGrid for use with BuildStream

# THINGS TO BE AWARE OF

- Protobufs might make you sad
  - Pin "known-good" versions to avoid segfaults
  - Public code, private development
- Infrastructure experts required
  - BuildBarn, Buildfarm and BuildGrid all require some work to deploy
- Cache expiry
  - For best results, use the cache as a cache

# REAPI WISHLIST

## Clients:

- More build tools supporting REAPI
  - sccache ([link](#))
  - Distro packaging tools
- Improve BuildStream experience for developers
  - Faster rebuilds, perhaps with Recc inside Bst

## Infrastructure:

- Wider support
  - Artifactory supporting REAPI caching ([issue](#))
- Easier deployment and management
- Better documentation

Find out more about the REAPI: <https://github.com/bazelbuild/remote-apis>

# BUILDSTREAM

- Integration tool comparable to Buildroot or BitBake
- Open source, Apache Foundation project
- Designed around strong caching
- Control all inputs to build process and sandbox build process
- Version 2.0 (releasing this month) designed around REAPI



# BUILDSTREAM - EXAMPLE PROJECT

elements/base.bst

```
1 kind: import
2 description: |
3
4     Alpine Linux base runtime
5
6 sources:
7 - kind: tar
8
9     # This is a post doctored, trimmed down system image
10    # of the Alpine linux distribution.
11    #
12    url: alpine:integration-tests-base.v1.x86_64.tar.xz
13    ref: 3eb559250ba82b64a68d86d0636a6b127aa5f6d25d3601a79f79214dc9703639
```

elements/hello.bst

```
1 kind: autotools
2 description: |
3
4     Hello world example from automake
5
6 variables:
7
8     # The hello world example lives in the doc/amhello folder.
9     #
10    # Set the %{command-subdir} variable to that location
11    # and just have the autotools element run its commands there.
12    #
13    command-subdir: doc/amhello
14
15 sources:
16 - kind: tar
17   url: gnu:automake-1.16.tar.gz
18   ref: 80da43bb5665596ee389e6d8b64b4f122ea4b92a685b1dbd813cd1f0e0c2d83f
19
20 depends:
21 - base.bst
```

# BUILDSTREAM - EXAMPLE PROJECT

elements/base.bst

```
1 kind: import
2 description: |
3
4     Alpine Linux base runtime
5
6 sources:
7 - kind: tar
8
9     # This is a post doctored, trimmed down system image
10    # of the Alpine linux distribution.
11    #
12    url: alpine:integration-tests-base.v1.x86_64.tar.xz
13    ref: 3eb559250ba82b64a68d86d0636a6b127aa5f6d25d3601a79f79214dc9703639
```

elements/hello.bst

```
1 kind: autotools
2 description: |
3
4     Hello world example from automake
5
6 variables:
7
8     # The hello world example lives in the doc/amhello folder.
9     #
10    # Set the %{command-subdir} variable to that location
11    # and just have the autotools element run its commands there.
12    #
13    command-subdir: doc/amhello
14
15 sources:
16 - kind: tar
17   url: gnu:automake-1.16.tar.gz
18   ref: 80da43bb5665596ee389e6d8b64b4f122ea4b92a685b1dbd813cd1f0e0c2d83f
19
20 depends:
21 - base.bst
```

# BUILDSTREAM - EXAMPLE PROJECT

elements/base.bst

```
1 kind: import
2 description: |
3
4     Alpine Linux base runtime
5
6 sources:
7 - kind: tar
8
9     # This is a post doctored, trimmed down system image
10    # of the Alpine linux distribution.
11    #
12    url: alpine:integration-tests-base.v1.x86_64.tar.xz
13    ref: 3eb559250ba82b64a68d86d0636a6b127aa5f6d25d3601a79f79214dc9703639
```

elements/hello.bst

```
1 kind: autotools
2 description: |
3
4     Hello world example from automake
5
6 variables:
7
8     # The hello world example lives in the doc/amhello folder.
9     #
10    # Set the %{command-subdir} variable to that location
11    # and just have the autotools element run its commands there.
12    #
13    command-subdir: doc/amhello
14
15 sources:
16 - kind: tar
17   url: gnu:automake-1.16.tar.gz
18   ref: 80da43bb5665596ee389e6d8b64b4f122ea4b92a685b1dbd813cd1f0e0c2d83f
19
20 depends:
21 - base.bst
```

# BUILDSTREAM - EXAMPLE PROJECT

elements/base.bst

```
1 kind: import
2 description: |
3
4     Alpine Linux base runtime
5
6 sources:
7 - kind: tar
8
9     # This is a post doctored, trimmed down system image
10    # of the Alpine linux distribution.
11    #
12    url: alpine:integration-tests-base.v1.x86_64.tar.xz
13    ref: 3eb559250ba82b64a68d86d0636a6b127aa5f6d25d3601a79f79214dc9703639
```

elements/hello.bst

```
1 kind: autotools
2 description: |
3
4     Hello world example from automake
5
6 variables:
7
8     # The hello world example lives in the doc/amhello folder.
9     #
10    # Set the %{command-subdir} variable to that location
11    # and just have the autotools element run its commands there.
12    #
13    command-subdir: doc/amhello
14
15 sources:
16 - kind: tar
17   url: gnu:automake-1.16.tar.gz
18   ref: 80da43bb5665596ee389e6d8b64b4f122ea4b92a685b1dbd813cd1f0e0c2d83f
19
20 depends:
21 - base.bst
```

# BUILDSTREAM - EXAMPLE PROJECT

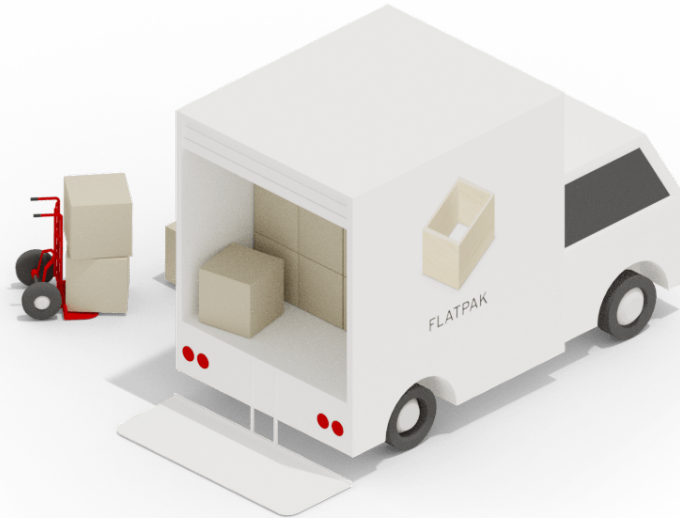
project.conf

```
1 # Unique project name
2 name: autotools
3
4 # Minimum required BuildStream version
5 min-version: 2.0
6
7 # Subdirectory where elements are stored
8 element-path: elements
9
10 # Define some aliases for the tarballs we download
11 aliases:
12   alpine: https://bst-integration-test-images.ams3.cdn.digitaloceanspaces.com/
13   gnu: http://ftpmirror.gnu.org/gnu/automake/
14
15 plugins:
16 - origin: pip
17   package-name: buildstream-plugins
18   elements:
19     - autotools
```

<https://github.com/apache/buildstream/tree/master/doc/examples/autotools>

# CASE STUDY 1

Freedesktop SDK and Runtime



freedesktop-sdk

# CASE STUDY: FREEDESKTOP SDK

- Stable base runtime for most Flatpak apps.
- Integrates > 600 open source components across 4 architectures with BuildStream.
- Largely volunteer-powered.
- Driven by Gitlab CI

The screenshot shows the GitLab interface for the 'freedesktop-sdk' project. The left sidebar contains navigation links: Project information, Repository, Issues (177), Merge requests (55), Requirements, CI/CD, Pipelines (selected), Editor, Jobs, Schedules, Test Cases, Security & Compliance, Deployments, Monitor, Infrastructure, Packages & Registries, Analytics, Wiki, Snippets, and Settings. The main content area displays the 'Pipelines' tab with a table of pipeline runs. The table has columns for Status, Pipeline ID, Triggerer, Commit, Stages, and Duration. The first five rows show 'passed' status, while the next three show 'passed' with a warning icon. The last row shows a 'passed' status with a warning icon. The table is filtered by 'All' and shows 1,000+ pipelines. Buttons for 'Clear runner caches', 'CI lint', and 'Run pipeline' are visible at the top right of the table.

Status	Pipeline	Triggerer	Commit	Stages	Duration
passed	#324900603		update/incl... ↪ 43c09f27 Update elements/include/li...	✓ ✓	10:38:41 18 hours ago
passed	#324900497		update/comp... ↪ 15d4e445 Update elements/compon...	✓ ✓	04:30:51 1 day ago
passed	#324888230		update/comp... ↪ 76262ae8 Update elements/compon...	✓ ✓	02:22:43 1 day ago
passed	#324885595 Scheduled		release/20... ↪ 0e474662 Merge branch 'abderahim/...	! ✓	00:50:51 1 day ago
passed	#324867178 Scheduled latest		19.08 ↪ 00ee2610 Merge branch 'nanonym/u...	! ✓ »	01:00:36 1 day ago
passed	#324866541 Scheduled		master ↪ 275eef83 Update elements/include/li...	! ✓	01:06:17 1 day ago
passed	#324834062		update/boot... ↪ bf16f0aa Update elements/bootstra...	✓ ✓	09:15:37 23 hours ago
passed	#324833975		master ↪ 275eef83 Update elements/include/li...	✓ ✓ ! ✓ ✓ ✓	09:49:00 23 hours ago

# CASE STUDY: FREEDESKTOP SDK

## Benefit from using BuildStream and REAPI:

- **Shared cache server** provided by BuildBox (BuildGrid)
- CI pushes all built elements to shared cache
- "Full rebuild" is almost never necessary
- Component updates automated
- Downstream projects benefit from upstream cache

✓ passed Pipeline #632060534 triggered 7 hours ago by Freedesktop SDK Merge Bot

**Update elements/components/python3-networkx.bst to 2.8.6**

Updates elements/components/python3-networkx.bst to 2.8.6  
Previous version was 2.8.5

Pipeline Needs Jobs 11 Tests 0

Group jobs by Stage Job dependencies

Bootstrap	Flatpak	Vm
bootstrap_ppc64	app_aarch64	desktop_efi_vm_image_x86_64
	app_i686	minimal_efi_vm_image_x86_64
	app_riscv64	minimal_efi_vm_image_x86_64 - passed
	app_x86_64	minimal_systemd_vm_x86_64
	pylint	

# CASE STUDY 2

Building for safety with DCS



Image: [Jason Armstrong](#), used under CC BY-NC-SA license

# CASE STUDY: DCS

DCS = "Deterministic Construction Service"

Qualified tool integration for building complex software (e.g. Linux-based operating systems) for safety-critical systems

A design pattern for safely building software.

# CASE STUDY: DCS

DCS = "Deterministic Construction Service"

Qualified tool integration for building complex software (e.g. Linux-based operating systems) for safety-critical systems

A design pattern for safely building software.

Repeatable builds are safe builds:

- Re-run of DCS process reproduces exactly the same binary fileset
- Reproducibility is shown to be independent of the specific instantiation of DCS

# CASE STUDY: DCS

DCS = "Deterministic Construction Service"

Qualified tool integration for building complex software (e.g. Linux-based operating systems) for safety-critical systems

A design pattern for safely building software.

Repeatable builds are safe builds:

- Re-run of DCS process reproduces exactly the same binary files
- Reproducibility is shown to be independent of the specific instantiation of DCS

Allows us to:

- Upgrade or modify tools used to construct safety-critical software and verify that these changes have no impact on the output binaries
- Perform fine-grained impact analysis on the constructed software when updating source and dependencies

# CASE STUDY: DCS

DCS = "Deterministic Construction Service"

Qualified tool integration for building complex software (e.g. Linux-based operating systems) for safety-critical systems

A design pattern for safely building software.

Repeatable builds are safe builds:

- Re-run of DCS process reproduces exactly the same binary files
- Reproducibility is shown to be independent of the specific instantiation of DCS

Allows us to:

- Upgrade or modify tools used to construct safety-critical software and verify that these changes have no impact on the output binaries
- Perform fine-grained impact analysis on the constructed software when updating source and dependencies

Reference implementation uses BuildStream.

# CASE STUDY: DCS

Qualified using ISO 26262 safety standard.



More information about safe software:

- see Paul Albertella speak on **Friday, 11.45**
- get involved in [ELISA](#)

# HAPPY INTEGRATING

Please invest in your build + integration pipelines, and...

Check out **BuildStream 2.0** at <https://buildstream.build>



Sam Thursfield

OSSEU 2022

